DA6401_Assigment 1 Report.

Sonalika Singh (MA23C044)

Sonalika_Singh

Created on March 8 | Last edited on March 17

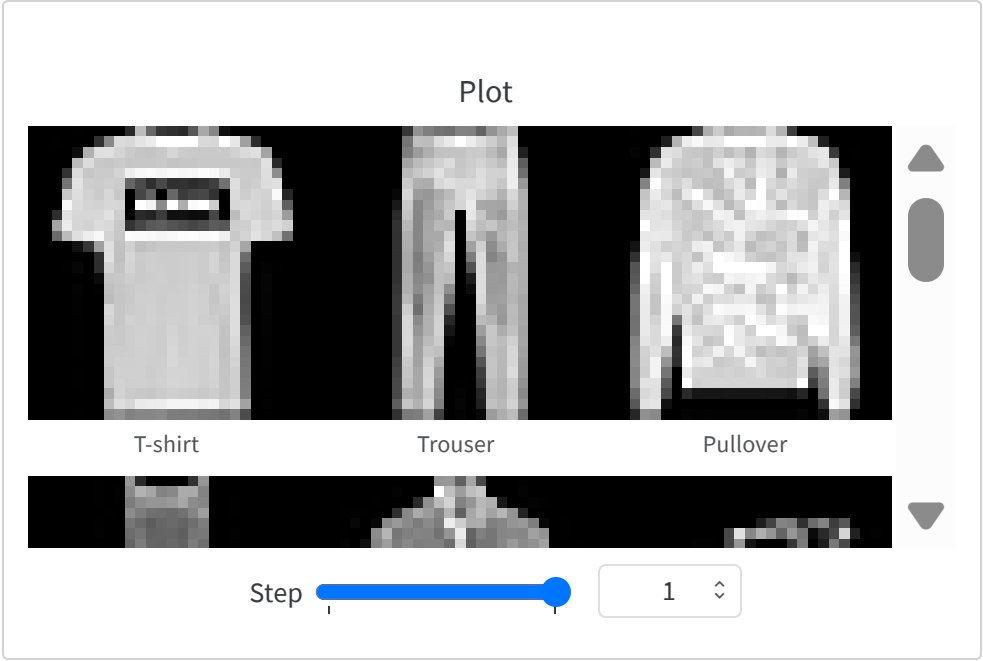
Problem Statement

In this assignment you need to implement a feedforward neural network and write the backpropagation code for training the network. We strongly recommend using numpy for all matrix/vector operations. You are not allowed to use any automatic differentiation packages. This network will be trained and
tested using the Fashion-MNIST dataset. Specifically, given an input image (28 x 28 = 784
pixels) from the Fashion-MNIST dataset, the network will be trained to classify the image
into 1 of 10 classes.

Github repository link :
https://github.com/Sonalikasingh17/DA6401_Assignment-01

Question 1 (2 Marks)

Download the fashion-MNIST dataset and plot 1 sample image for each class. Use "from keras.datasets import fashion_mnist" for getting the fashion mnist dataset.



Question 2 (10 Marks)

Neural Network class has been implemented which takes in the training dataset, testing dataset, validation dataset and hyperparameters of the neural network including the number of hidden layers and hidden neurons.

Forward and backward propagation algorithms have been implemented and tested for various optimizers. An object oriented approach was followed with all the hyperparameters and model parameters(weights and biases) being the attributes of the Neural Network class.

Code uploaded to Github.

Question 3 (18 Marks)

Implemented all the optimizer as was asked in the question (except NADAM).
The back propagation algorithm was written for both Cross entropy and Mean Squared error loss functions in a parametrized manner.
For more details check the readme.

Question 4 (10 Marks)

The highest accuracy that we have achieved is 89.15 % on validation data.

Strategy for hyper-parameter tuning

Used the sweep functionality provided by wandb to find the best values for the hyper parameters listed below. The standard train/test split of fashion_mnist has been used with 60000 training images and 20000 testing images and labels. 10% of the randomly shuffled training data was kept aside as validation data for hyperparameter search (6000 images).

Following are the hyperparameter options for which wandb sweep was performed.
number of epochs: 5, 10

number of hidden layers: 3, 4, 5

size of every hidden layer: 32, 64, 128

weight decay (L2 regularization): 0, 0.0005, 0.5

learning rate: 1e-3, 1 e-4

optimizer: sgd, momentum, nesterov, rmsprop, adam

batch size: 16, 32, 64

weight initialisation: random, Xavier

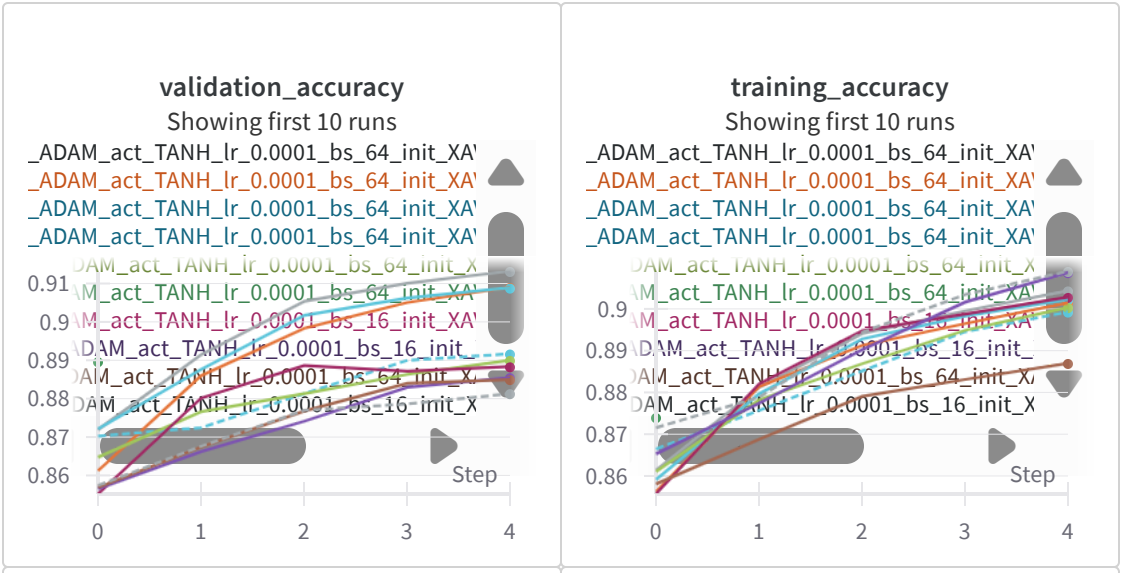
activation functions: sigmoid, tanh, ReLU

these are the number of parameters which were given, since there are lots of combination possible.
That's why I made a sweep Bayesian_hyperparam_tuning used Bayesian optimisation.
From this Bayesian sweep, I got a maximum validation accuracy of 89.15%.

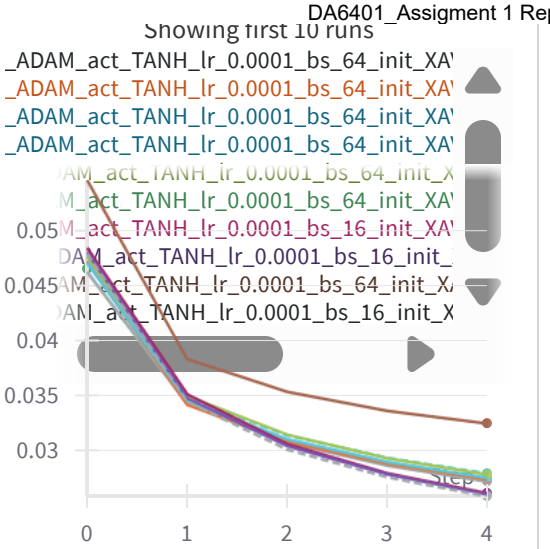
Bayes where we aggressively eliminated parameters at each sweep and found out that Bayes sweep works well if there is lower number of parameters where every combination is good. If there is any bad parameter there it affects other parameters as with which the bad parameter was paired.

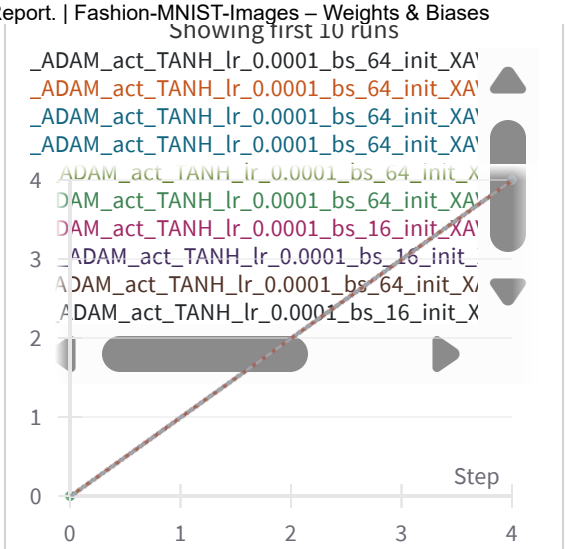
Configuration for best model:

Model No.	Hidden Layers	Hidden Neurons	Activation	Optimizer	Learning Rate	Weight Decay	Accura
1	3	128	tanh	Adam	0.0001	0.0005	89.15 %



loss epoch


Showing first 10 runs
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
0.05
0.045
0.04
0.035
0.03
0
1
2
3
4


Showing first 10 runs
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
_ADAM_act_TANH_lr_0.0001_bs_64_init_XA
4
3
2
1
0
Step
0
1
2
3
4

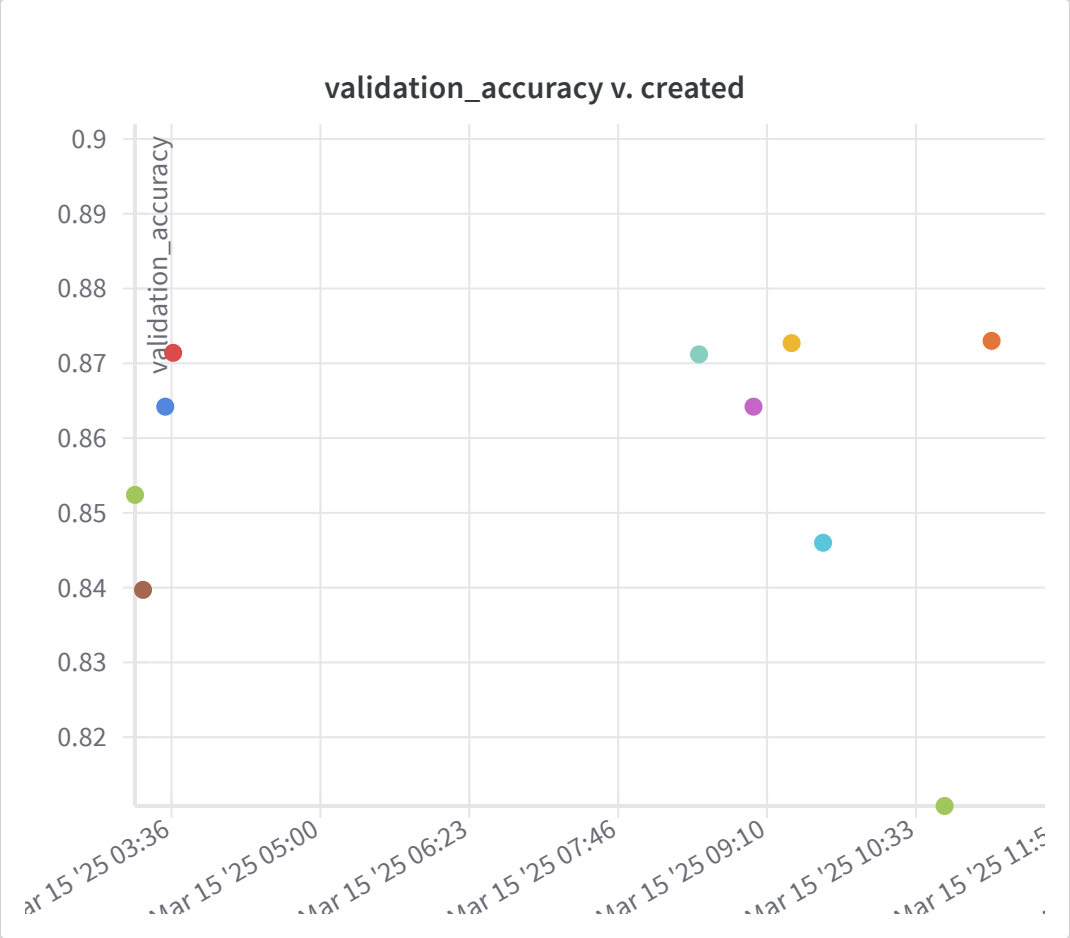
Run 138

Question 5 (5 marks)

The accuracy plot is presented below.

Plot of validation_accuracy versus created timestamp.

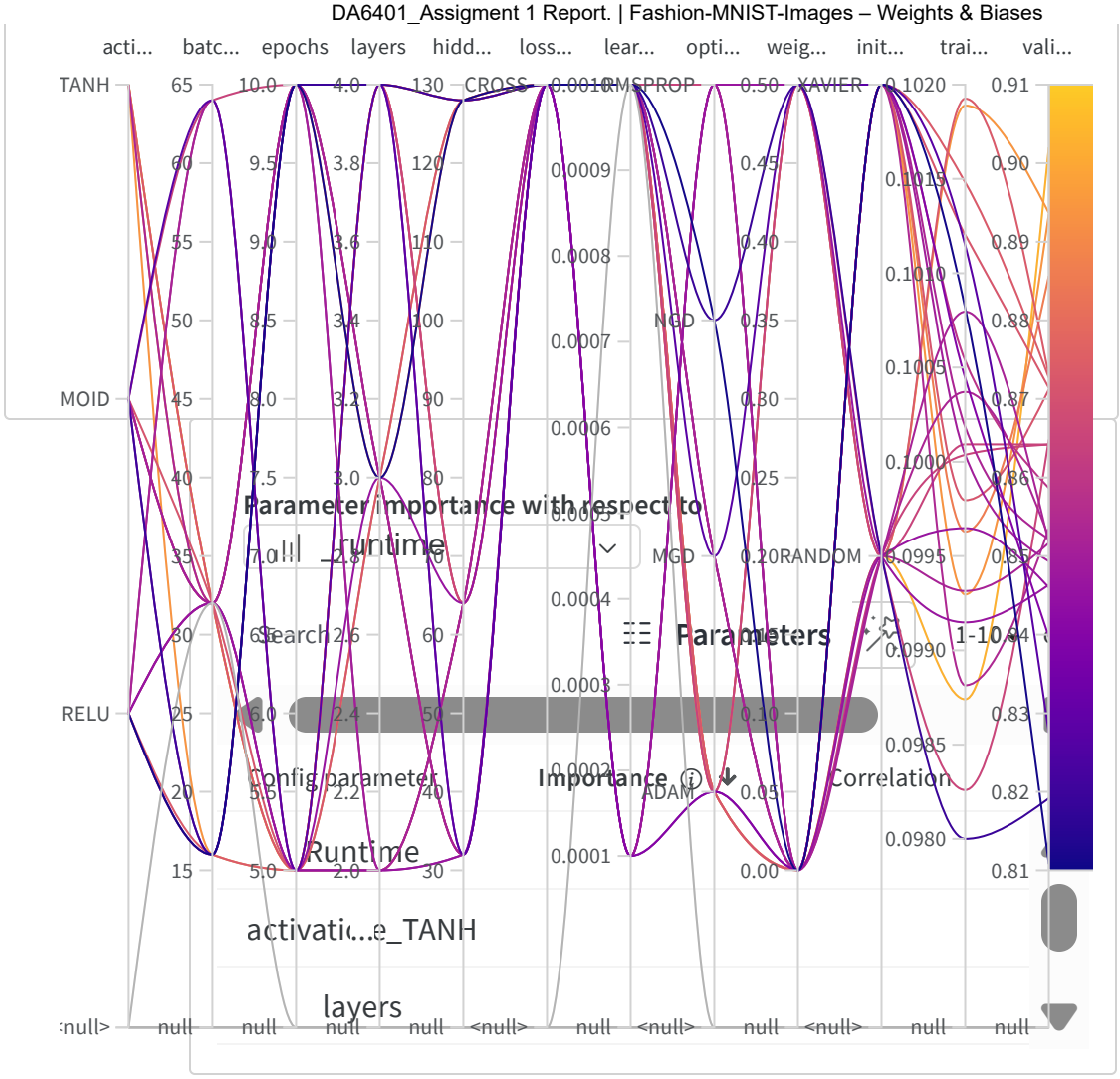
since,wandb automatically generates this plot which summarises the test accuracy of all the models that I tested.



Question 6 (20 Marks)

Observations

- ADAM is better at getting good accuracy results.
- Lower learning rate is better.
- Weight decay parameter of range around .3-.4 is giving good results.
- Stochastic gradient algorithms get stuck before going to a good accuracy value. This happens mainly because the gradient value becomes low when it nears the optimum point.
- Xavier initialization is better than randomly initializing the weights.
- The batch size we use depends on the number of epochs we can afford . While higher batch size gives a better stability of gradient, it also lets us take fewer steps.
- Learning rate, number of hidden neurons, number of hidden layers, optimizer(ADAM) and activation(TANH) play an important role in determining the validation accuracy. This is inferred from the correlation summary.
- While learning rate, number of hidden neurons, hidden layers have a positive correlation with validation accuracy, weight decay seems to have a negative impact in the validation accuracy for most cases.



Question 7 (10 Marks)

For the best model identified above, report the accuracy on the test set of fashion_mnist and plot the confusion matrix as shown below.
Best model found

Model No.	Layers	Hidden Neurons	Activation	Optimizer	Learning Rate	Weight Decay	Accura
1	2	128	tanh	Adam	0.0001	0.0005	87.98 %

Note:

Due to some reasons I am not able to visualize my confusion matrix here.

Pasted a link to visualize the confusion matrix. Please go through it, and asking for an apology. Sorry, for the inconvienence.

Test_Confusion_Matrix

https://wandb.ai/singhsonalika5-indian-institute-of-technology-madras/Fashion_MNIST_Images/reports/undefined-25-03-17-23-11-44---VmlldzoxMTg0MTAyMA?accessToken=c26arrmh5qbi3yjb0sm9abx5ccqwnoz40oqmrzqifn88imtinqgya4lcacunvlwc

Train confusion Matrix

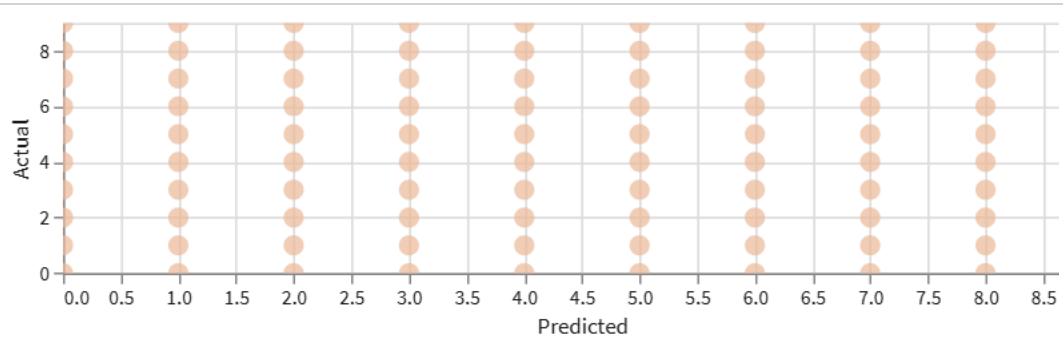
https://wandb.ai/singhsonalika5-indian-institute-of-technology-madras/Fashion_MNIST_Images/reports/undefined-25-03-17-23-17-37---VmlldzoxMTg0MTEzNA?accessToken=7s54w074xlltoc7cm8s8ge5kq6o01d37rdpnnzosg5bht53hceuroh1kc3ua5r7m

confusion_matrix

Step 0

Test_Confusion_Matrix

Predicted	Actual	Count
0	0	868
0	1	2
0	2	10
0	3	17
0	4	2
0	5	1



1 of 2

my_bar_chart_id_table

Test_Confusion_Matrix

Configuration	Test accuracy
test_pred_best	0.8794

Run set 2 2

Question 8 (5 Marks)

To compare squared error and cross entropy loss function for the given problem, I have used the best model configuration obtained

for cross entropy loss and for Mean squared error loss function which is as follows:

Number of Hidden Layers - 3

Number of Hidden Neurons - 128

L2 Regularisation - 0.005

Activation - Tanh

Initialisation - Xavier

Optimiser - ADAM

Learning Rate - 0.0001

Batch size - 64

The back propagation function implemented in the code allows one to choose between the loss function types at the configuration stage itself.

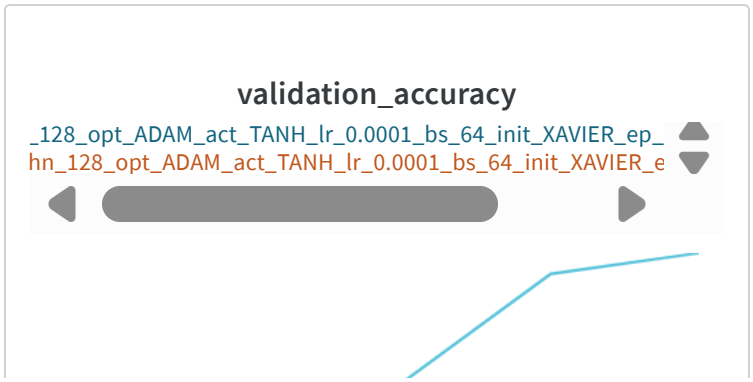
Ran the same model with 2 different loss function.

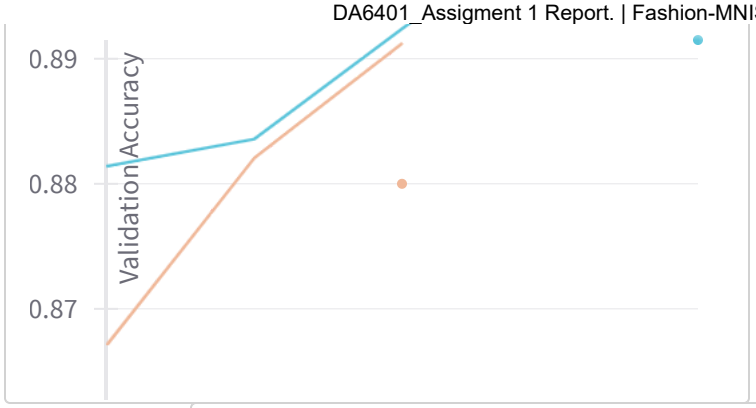
After the run cross entropy gave- 89.15 % accuracy

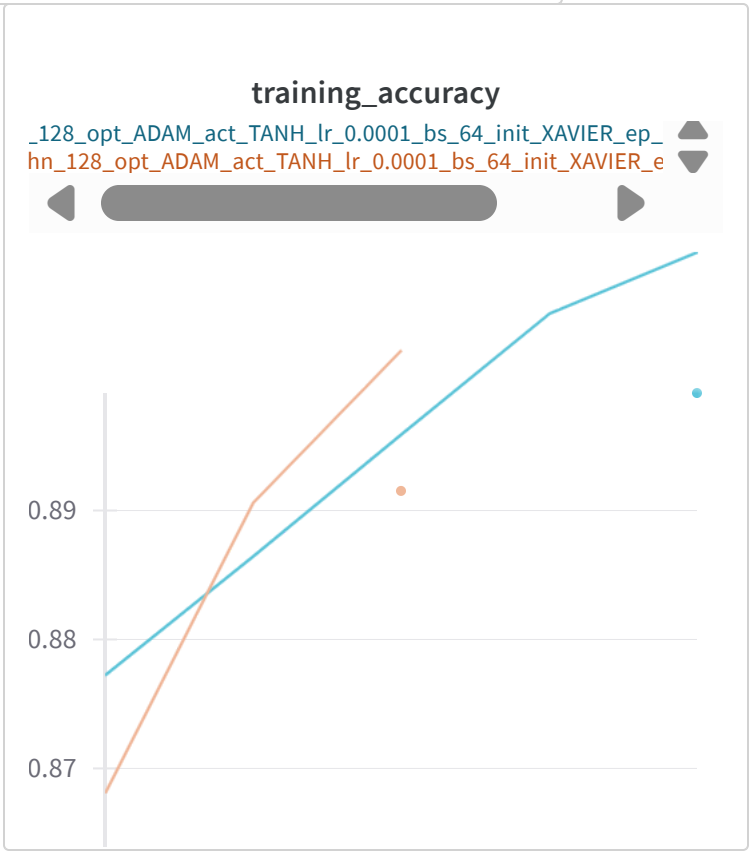
Squared Error loss gave-88 %

conclude that squared error loss is not properly capturing the picture of a probability distribution.

Conclusion: Use cross entropy loss.







Question 9 (10 Marks)

https://github.com/Sonalikasingh17/DA6401_Assignment-01

[> Read Me]

(https://github.com/Sonalikasingh17/DA6401_Assignment-01/blob/main/README.md)

Question 10 (10 Marks)

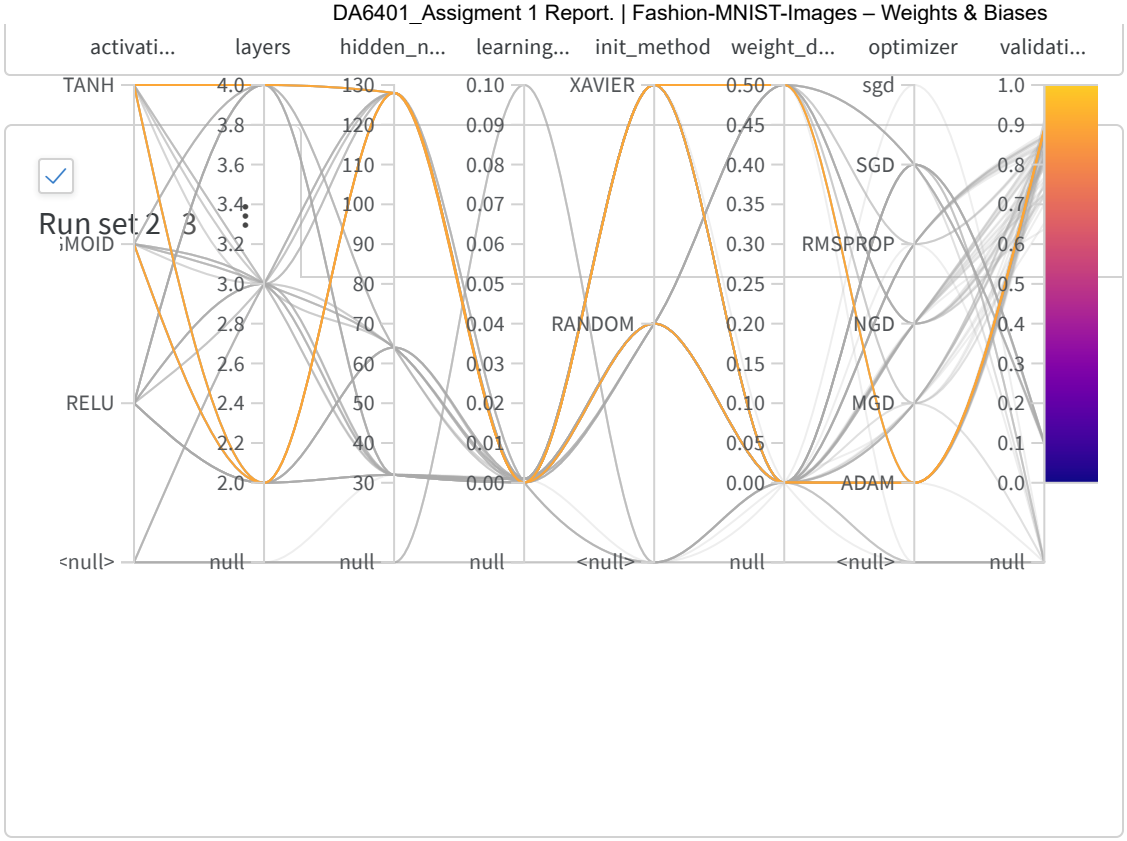
Used three of the best models that performed best for Fashion - MNIST data with out using data augmentation.

We felt that since MNIST is a simpler dataset , data augmentation will be unnecessary for it.

The three models we use :-

Model No.	Layers	Hidden Neurons	Activation	Optimizer	Learning Rate	Weight Decay	Accura
1	2	128	tanh	Adam	0.0001	0.0005	89.15 %
2	4	128	tanh	Adam	0.0001	0.5	88.89 %
3	2	128	sigmoid	Adam	0.0001	0.0005	88.81 %

Conclusion - Whatever works for Fashion MNIST , works better for MNIST.



Self Declaration

This assignment was solved individually.

MA23C044: (100% contribution)

Implementing the data preprocessing pipeling including one hot encoding of the output labels.

Implementing sgd, nesterov, rmsprop, adam optimisers

Implementing Random and Xavier initialisations.

Implementing the feed forward neural network class with a generalised structre for forward and backward propagation algorithm

Setting up the code base for wandb and configuration of sweeps.

Make a python file to run bayesian sweeps.

Plotting the confusion matrix.

Analysing the parallel co-ordinates plot and writing inferences.

I, Sonalika Singh, swear on my honour that the above declaration is correct.

Created with ❤️ on Weights & Biases.

https://wandb.ai/singhsonalika5-indian-institute-of-technology-madras/Fashion-MNIST-Images/reports/DA6401_Assigment-1-Report---VmlldzoxMTY5Nzc5Nw